41/prts

GAME SYSTEM AND IMAGE GENERATION METHOD

TECHNICAL FIELD

The present invention relates to a game system, program and image generation method.

BACKGROUND ART

There is known an image generating system which can generate an image as viewed within a virtual three-dimensional or object space from a given viewpoint. Such a system is very popular since one can experience a so-called virtual reality through it. Now considering an image generating system for playing a racing game, a player can enjoy a three-dimensional shooting game by manipulating a racing car (or object) to run in an object space and to compete against racing cars which are manipulated by other players and computer.

In such a game system, it is desirable that a transformation known as gamma correction relating to an image is performed to correct the non-linear characteristics of a monitor (or display section).

There are known two techniques for realizing such a gamma correction.

A first technique has provided a gamma-correction lookup table (LUT) on a main memory 802, as shown in Fig. 1A. CPU 800 (or a software running on the CPU) reads the color information (RGB) of the respective pixels in an original image out of a frame buffer 808 in VRAM 806. The gamma-correction LUT is then

referred to based on the read color information to obtain the gamma-corrected color information. Next, the gamma-corrected color information so obtained is written in the corresponding pixel in the frame buffer. Such a procedure will be repeated for all the pixels in the original image.

On the other hand, a second technique provides a gamma-correction circuit 814 located downstream of a drawing processor 812, as shown in Fig. 1B, which is operated under control of the CPU 810 and designed to realize the gamma correction in hardware. The gamma-correction circuit 814 performs the gamma correction relative to the color information generated by the drawing processor 812, the gamma-corrected color information being then outputted toward a monitor 816.

However, the first technique of Fig. 1A requires a software running on the CPU 800 for performing all the procedures of reading the color information out of the frame buffer 808, referring the gamma-correction LUT 804, reading the color information out of the gamma-correction LUT 804 and writing the read color information back to the frame buffer 808. Thus, the process will not be executed at any higher speed and it is difficult to complete the gamma correction for all the pixels in a display screen within one frame. Moreover, the processing load in the CPU 800 becomes very heavy. This adversely affects any other processing.

On the other hand, the second technique of Fig. 1B can realize the gamma correction with a higher speed since the gamma-correction circuit 814 used is of dedicated hardware.

Therefore, the gamma correction for all the pixels in a display screen can easily be accomplished within one frame. Since the processing load of the CPU 810 is reduced, furthermore, the other processing will not adversely be affected by the gamma correction.

However, the second technique of Fig. 1B separately requires the gamma correction circuit 814 which is of dedicated hardware. Thus, the game system will be increased in scale, leading to increase of the manufacturing cost.

In domestic game systems, particularly, the severe reduction of cost is required to make the products more popular. Thus, most domestic game systems have not included such a hardware gamma-correction circuit as shown in Fig. 1B. To realize the gamma correction in the domestic game systems, they cannot but take such a first technique as shown in Fig. 1A.

Since it is difficult in the first technique to complete the gamma correction for the entire screen within one frame as described, however, it will adversely affect the other processing. As a result, the domestic game systems could not but abandon the adoption of the gamma correction itself.

The game systems of the prior art have a further problem described below.

An image generated by any prior art game system was not focused depending on the distance from a viewpoint, unlike the image viewed through human eyes. Thus, the image was represented such that all the objects in the image were focused.

However, such an image is so unnatural that the human

will never see in its daily life.

To provide more realistic images, it is therefore desirable that they are generated to have objects which are focused depending on the distance between the viewpoint and the objects, the direction of line-of-sight and so on. However, when defocused images are generated by computing the distance between each of the individual objects and the viewpoint in the game space to provide the necessary degree of defocusing for every object, the processing load will hugely be increased.

In a game system which is required to generate images corresponding to the viewpoint which varies in real time by using a limited hardware resource, it is important that such images being focused as if it were real views are generated with reduced processing load.

DISCLOSURE OF THE INVENTION

In view of the aforementioned problems, an objective of the present invention is to provide a game system, program and image generating method which can implement video filtering such as gamma correction with reduced processing load.

Another objective of the present invention is to provide a game system, program and image generating method which can generate more realistic images with reduced processing load. More particularly, it is to provide a game system, program and image generating method which can generate an image focused like a real view with reduced processing load.

To this end, the present invention provides a game system

which generates an image, comprising: means which sets image information of an original image as an index number in a lookup table for index color texture-mapping; and means which transforms the image information of the original image by performing index color texture-mapping on a virtual object by using the lookup table in which the image information of the original image is set as the index number. The present invention also provides a computer-usable information storage medium comprising a program for a computer to realize the above means. The present invention further provides a computer-usable program (including a program embodied in a carrier wave) comprising a processing routine for a computer to realize the above-described means.

According to the present invention, the lookup table in which the image information of the original image is set as an index number is used to perform the index color texture-mapping on a virtual object for transforming the image information of the original image. According to the present invention, thus, the index color texture-mapping function originally possessed by the game system (or image generating system) can effectively be used to perform the transformation of the image information of the original image. The video filtering processes such as gamma correction can be executed with higher speed without additional hardware. Moreover, the transformation of image information of the entire display screen can more easily be performed.

The image information of the original image may be one

drawn in a drawing region (such as a frame buffer or an additional buffer) and contain color information, an alpha (α) value, depth value or the like. The transformation of the image information of the original image is not limited to gamma correction, and any of various other types of transformation can be considered. The transformed image information obtained by setting the image information of the original image as an index number in the lookup table is not limited to the color information. The virtual object may be one or more primitive surfaces (such as polygons or free-form surfaces).

In the game system, information storage medium or program of the present invention, the virtual object may be a polygon having a size equal to a size of a display screen.

In this configuration, the image information of the original image of the entire display screen can be transformed by one or several times of texture mapping.

In the game system, information storage medium or program of the present invention, the virtual object may be a polygon having a size equal to a size of a block obtained by dividing a display screen into blocks.

In this configuration, the region for the drawing of the virtual object can be reduced in size to save the usable capacity of a storage section.

In the game system, information storage medium or program of the present invention, the lookup table may be used to perform gamma correction, negative/positive inversion, posterization, solarization, binarization, monotone filtering

or sepia filtering on the image information of the original image.

In this configuration, an image obtained by performing various image effects on the original image can be generated. This improves the variety in the generated image. Note that the transformation of image information according to the present invention is not limited to gamma correction, negative/positive inversion, posterization, solarization, binarization, monotone filtering and sepia filtering.

In the game system, information storage medium or program of the present invention, one of color components of color information in the image information of the original image may be set as the index number in the lookup table for the transformation of the color information; and the game system, information storage medium or program may further comprise means (or a program or a processing routine for a computer to realize the means) which performs masking on other color components of the transformed color information to avoid being drawn in the drawing region.

In this configuration, while using the lookup table for index color texture-mapping which outputs a plurality of values relative to one input value, image transformation such as the gamma correction which outputs one value relative to one input value can be performed.

The game system, information storage medium or program of the present invention may further comprise means (or a program or processing routine for a computer to realize the

means) which blends:

transformed color information obtained by setting the K-th color component of the color information in the image information of the original image as the index number in the lookup table;

transformed color information obtained by setting the L-th color component of the color information as the index number in the lookup table; and

transformed color information obtained by setting the M-th color component of the color information as the index number in the lookup table.

This configuration can implement image transformation in which an image having color information obtained by blending K-, L- and M-th color components is generated relative to the input of the K-th color component, for example.

In the game system, information storage medium or program of the present invention, an alpha value corresponding to the image information of the original image may be generated by the transformation of the image information of the original image.

In this configuration, if the image information of the original image is a depth value, an alpha value of each pixel can be set at a value corresponding to the depth value of each pixel in the original image. Thus, an original image can be blended with a defocused image of the original image based on the alpha values set for the respective pixels. This also enables representation of the depth of field.

By utilizing the generated alpha values, the masking or the like corresponding to the values of the image information of the original image can be carried out.

In the game system, information storage medium or program of the present invention, a depth value in the image information of the original image may be set as the index number in the lookup table.

In this configuration, various types of information can be provided as the image information to be set as the index number in the lookup table.

The present invention further provides a game system which generates an image, comprising: means which sets a depth value of each pixel of an original image as an index number in a lookup table for index color texture-mapping; means which sets an alpha value of each pixel to a value corresponding to the depth value of each pixel of the original image by performing index color texture-mapping on a virtual object by using the lookup table in which the depth value of each pixel of the original image is set as the index number; and means which blends the original image with a defocused image of the original image based on the alpha value of each pixel. The present invention further provides a computer-usable information storage medium comprising a program for a computer to realize the above means. The present invention further provides a computer-usable program (including a program embodied in a carrier wave) comprising a processing routine for a computer to realize the above-described means.

According to the present invention, the lookup table in which the depth value for each pixel in the original image is set as an index number is used to perform index color texture-mapping on a virtual object and to set an alpha value for each pixel to a value corresponding to the depth value for each pixel in the original image. Thus, the present invention can effectively use the index color texture-mapping function originally possessed by the game system (or image generating system) for transforming the depth value into the alpha value. The transformation of the depth value can easily be carried out for the entire display screen with increased speed and without addition of any hardware.

According to the present invention, an original image can be combined with a defocused image of the original image based on the alpha values set at values corresponding to the depth values for pixels in the original image. Therefore, the rate of blending the original image with the defocused image or other factors can be changed depending on the depth value, thus enabling representation of the depth of field or the like.

Note that the alpha value is information stored in association with each pixel and may be one other than the color information, for example. The depth value may be larger as the distance from the viewpoint (or a virtual camera) is shorter or longer. The technique of the defocused image which is blended with the original image may be any of various techniques. The blending process using an alpha values is not limited to the alpha(α)-blending in narrow sense.

In the game system, information storage medium or program of the present invention, the depth value of each pixel of the original image may be transformed into a second depth value formed of lower bits I to J which are positioned lower than the most significant bit of the depth value; and the second depth value may be set as the index number in the lookup table for index color texture-mapping.

In this way, the effective number of partitions for the alpha value (or the effective number of steps for the threshold value of the depth value) can be increased. For example, the degree of defocusing in an object located close to the focus (or a gazing point) of a virtual camera can be controlled with improved accuracy by multi-step threshold values. Therefore, the quality of the generated image can be improved.

In the game system, information storage medium or program of the present invention, the second depth value may be clamped into a given value depending on a bit value other than the bits I to J in the depth value.

If a bit other than the bits I to J in the depth value becomes 1, this configuration can generate a consistent image. The given value may be any of various values such as the maximum or minimum values of the second depth value, or a value in which a train of higher bits of the second depth value set to 1 and so on.

In the game system, information storage medium or program of the present invention, the depth value may be set as an index number in a lookup table for index color

texture-mapping; and the depth value may be transformed into the second depth value by performing index color texture-mapping on a virtual object by using the lookup table.

In this configuration, a variety of transformation processes, including clamping of the second depth value into a given value, can be implemented with a reduced processing load in which only the transforming properties of the lookup table are changed.

In the game system, information storage medium or program of the present invention, bits M to N in the depth value may be set as an index number in a first lookup table for index color texture-mapping; the depth value may be transformed into a third depth value by performing index color texture-mapping on a virtual object by using the first lookup table; bits K to L (where $K \geqq I \geqq L > M \geqq J \geqq N$) in the depth value may be set as an index number in a second lookup table for index color texture-mapping; the depth value may be transformed into a fourth depth value by performing index color texture-mapping on a virtual object by using the second lookup table; and the third and fourth depth values may be used to determine the second depth value.

In such a manner, if there is a restriction by which only a train of bits within a given range of the depth value (e.g., 0 to 7, 8 to 15, 16 to 23 or 24 to 31 bits) can be fetched, the second depth value formed of any bits between I and J in the depth value can be obtained. Thus, the effective number of steps of the threshold values of the depth value for partitioning the

alpha value can be increased to improve the quality of generated images.

In the game system, information storage medium or program of the present invention, the defocused image of the original image may be generated by setting the original image as a texture and shifting texture coordinates of a virtual object when the texture is mapped onto the virtual object by texel interpolation method.

In this configuration, a simplified process in which an original image is mapped on a virtual object by the texel interpolation method while shifting the texture coordinates can be used to defocus the original image.

The texel interpolation method is not limited to particular processes, but it is a process to obtain image information of a pixel by interpolating image information of a texel, and there is bilinear filtering or trilinear filtering, for example.

A virtual object may be primitive surfaces such as polygons or the like, or may be a three-dimensional object. Although it is desirable that the virtual object is not displayed on a screen, it may be displayed on the screen.

When a defocused image is generated by texture mapping using texel interpolation method, it is desirable that the texture coordinates are shifted by a value smaller than one texel. After texture mapping using texel interpolation method is performed by shifting the texture coordinates in a direction of a first shift, texture mapping using texel interpolation

method may be carried out again by shifting the texture coordinates in a direction of a second shift. Alternatively, a set of shifting in directions of a first shift and a second shift may be repeated several times.

The present invention further provides a game system which generates a game image for a domestic game, comprising: means which sets an adjustment data for adjusting display properties of a monitor based on operational data inputted by a player through a game controller; save means which saves the set adjustment data in a saved information storage device for storing personal data of the player; and means which performs transformation processing on image information of an original image based on the adjustment data obtained by adjusting the display properties or loaded from the saved information storage device. The present invention further provides a computer-usable information storage medium comprising a program for a computer to realize the above means. The present invention further provides a computer-usable program (including a program embodied in a carrier wave) comprising a processing routine for a computer to realize the above-described means.

According to the present invention, a player can use a game controller to set adjustment data for adjusting display properties of monitor (such as brightness, color density, tone or sharpness). Thus, the convenience for the player can be improved since it is not required that the player directly controls the monitor. The set adjustment data is saved in a saved information storage device, and image information of an

original image is transformed based on the adjustment data obtained by adjusting the display properties or loaded from the saved information storage device. Consequently, a game image can be displayed with optimum display properties for a game played by the player. Moreover, if a player views an image from other image source, a bad influence due to adjusted display properties of monitor can be prevented.

In the game system, information storage medium or program of the present invention, data of a control point of a free-form curve representing transformation properties of the image information may be saved in the saved information storage device as the adjustment data by the save means.

In this way, the capacity of the saved information storage device can be saved and saved capacity can be used for any other application.

BRIEF DESCRIPTION OF THE DRAWINGS

Figs. 1A and 1B illustrate first and second techniques for implementing gamma correction.

Fig. 2 is a block diagram of a game system according to this embodiment.

Fig. 3 illustrates the index color texture-mapping process.

Fig. 4 illustrates a technique of effectively using the index color texture-mapping LUT to transform an original image.

Figs. 5A and 5B exemplify game images generated according to this embodiment.

Fig. 6 illustrates a technique of dividing the original image into a plurality of blocks and texture mapping an image for each block onto the corresponding polygon having a size equal to that of each block using the LUT.

Figs. 7A and 7B exemplify the transformation property of the gamma correction and a gamma correction lookup table.

Figs. 8A and 8B exemplify the transformation property of the negative/positive inversion and a negative/positive inversion lookup table.

Figs. 9A, 9B and 9C exemplify the transformation properties of posterization, solarization and binarization.

Figs. 10A and 10B exemplify monotone (sepia) filtering LUTR and LUTG.

Fig. 11 exemplifies a monotone (sepia) filtering LUTB.

Fig. 12 illustrates a technique of obtaining a transformed image for the original image by masking other color components when a particular color component is set as an index number.

Fig. 13 illustrates a technique of blending color information obtained from LUTR, LUTG and LUTB to provide a transformed image for the original image.

Fig. 14 illustrates a technique of preparing an alpha(α)-plane by performing the texture mapping using the LUT.

Fig. 15 illustrates a technique of setting Z value at an index number in the LUT.

Fig. 16 illustrates a technique of setting an alpha value depending on the Z value and using the set alpha value to blend

the original image with a defocused image.

Figs. 17A and 17B exemplify an original image and its defocused image.

Fig. 18 illustrates a technique of setting an alpha value depending on the Z value.

Figs. 19A, 19B and 19C illustrate a technique of updating an alpha value of a deeper pixel from a virtual object by drawing the virtual object.

Fig. 20 illustrates a technique of blending an original image with its defocused image by transforming Z value into Z2 value and also transforming the Z2 value into an alpha value.

Fig. 21 illustrates a problem occurred when the Z2 value is formed by the high-order bits of the Z value including its most-significant bit.

Fig. 22 illustrates a technique of forming the Z2 value by the bits I to J lower than the most-significant bit of the Z value while clamping the Z2 value at a given value.

Fig. 23 illustrates a technique of transforming Z value into Z2 value using LUT.

Fig. 24 shows an example of LUT1 for transforming the bits 15-8 in the Z value.

Fig. 25 shows an example of LUT2 for transforming the bits 23-16 in the Z value.

Figs. 26A and 26B show an example of LUT3 for transforming a Z2 value into an alpha value and also exemplify the characteristic curve in that transformation.

Fig. 27 illustrates a clamping process.

Fig. 28 illustrates a bilinear filtering type texture mapping process.

Fig. 29 illustrates a technique of effectively using the bilinear filtering mode to generate a defocused image.

Fig. 30 also illustrates a technique of effectively using the bilinear filtering mode to generate a defocused image.

Figs. 31A and 31B illustrate the principle of generating a defocused image through the bilinear filtering type interpolation.

Figs. 32A and 32B also illustrate the principle of generating a defocused image through the bilinear filtering type interpolation.

Fig. 33 illustrates a problem in the prior art relating to the adjustment of brightness in a monitor.

Fig. 34 illustrates a technique of saving the data of the adjusted brightness in the monitor in a saved information storage device.

Fig. 35 is a flowchart illustrating the details of the process according to this embodiment.

Fig. 36 is a flowchart illustrating the details of the process according to this embodiment.

Fig. 37 is a flowchart illustrating the details of the process according to this embodiment.

Fig. 38 is a flowchart illustrating the details of the process according to this embodiment.

Fig. 39 is a flowchart illustrating the details of the process according to this embodiment.

Fig. 40 shows a structure of hardware by which this embodiment can be realized.

Figs. 41A, 41B and 41C show various system forms to which this embodiment can be applied.

BEST MODE FOR CARRYING OUT THE INVENTION

Preferred embodiments of the present invention will now be described in connection with the drawings.

1. Configuration

Fig. 2 shows a block diagram of a game system (or image generating system) according to this embodiment. In this figure, this embodiment may comprise at least a processing section 100. Each of the other blocks may take any suitable form.

The processing section 100 performs processing for control of the entire system, commands to the respective blocks in the system, game processing, image processing, sound processing and so on. The function thereof may be realized through any suitable hardware means such as various processors (CPU, DSP and so on) or ASIC (gate array or the like) or a given program (or game program).

An operating section 160 is used to input operational data from the player and the function thereof may be realized through any suitable hardware means such as a lever, a button, a housing or the like.

A storage section 170 provides a working area for the processing section 100, communication section 196 and others.

The function thereof may be realized by any suitable hardware means such as RAM or the like.

An information storage medium (which may be a computer-usable storage medium) 180 is designed to store information including programs, data and others. The function thereof may be realized through any suitable hardware means such as optical memory disk (CD or DVD), magneto-optical disk (MO), magnetic disk, hard disk, magnetic tape, memory (ROM) or the like. The processing section 100 performs processing in the present invention (or this embodiment) based on the information that has been stored in this information storage medium 180. In other words, the information storage medium 180 stores various pieces of information (programs or data) for causing a computer to realizing the means of the present invention (or this embodiment) which are particularly represented by the blocks included in the processing section 100.

Part or the whole of the information stored in the information storage medium 180 will be transferred to the storage section 170 when the system is initially powered on. The information stored in the information storage medium 180 may contain at least one of program code set for processing the present invention, image data, sound data, shape data of objects to be displayed, table data, list data, information for instructing the processing in the present invention, information for performing the processing according to these instructions and so on.

A display section 190 is to output an image generated

according to this embodiment and the function thereof can be realized by any suitable hardware means such as CRT, LCD or HMD (Head-Mount Display).

A sound output section 192 is to output a sound generated according to this embodiment and the function thereof can be realized by any suitable hardware means such as loudspeaker.

A saved information storage device (or portable information storage device) 194 stores the player's personal data (or data to be saved) and may take any suitable form such as memory card, portable game machine and so on.

A communication section 196 performs various controls for communication between the game system and any external device (e.g., host machine or other game system). The function thereof may be realized through any suitable hardware means such as various types of processors or communication ASIS or according to any suitable program.

The program or data for executing the means in the present invention (or this embodiment) may be delivered from an information storage medium included in a host machine (or server) to the information storage medium 180 through a network and the communication section 196. The use of such an information storage medium in the hose device (or server) falls within the scope of the invention.

The processing section 100 further comprises a game processing section 110, an image generation section 130 and a sound generation section 150.

The game processing section 110 performs processing such

as coin (or charge) reception, setting of various modes, game proceeding, setting of screen selection, determination of the position and rotation angle (about X-, Y- or Z-axis) of an object (or one or more primitive surfaces), movement of the object (motion processing), determination of the view point (or virtual camera' position) and visual-line angle (or rotational virtual camera angle), arrangement of the object within the object space, hit checking, computation of the game results (or scores), processing for causing a plurality of players to play in a common game space, various game computations including game-over and other processes, based on operational data from the operating section 160 and according to the personal data and game program from the portable information storage device 194.

The image generation section 130 performs image processing according to the commands or the like from the game processing section 110. For example, the image generation section 130 may generate an image within an object space as viewed from a virtual camera (or viewpoint), which image is in turn outputted toward the display section 190. The sound generation section 150 performs sound processing according to the commands and the like from the game processing section 110 for generating BGMs, sound effects, voices or the like which are in turn outputted toward the sound output section 192.

All the functions of the game processing section 110 and the image and sound generation sections 130, 150 may be realized by any suitable hardware means or according to the program.

Alternatively, these functions may be realized by both the hardware means and program.

The game processing section 110 further comprises a movement/action computation section 112, a adjustment information setting section 114 and a saving section 116.

The movement/action computation section 112 is to calculate the information of movement for objects such as motorcars and so on (positional and rotation angle data) and the information of action for the objects (positional and rotation angle data relating to the parts in the objects). For example, the movement/action computation section 112 may cause the objects to move and act based on the operational data inputted by the player through the operating section 160 and according to the game program.

More particularly, the movement/action computation section 112 may determine the position and rotational angle of the object, for example, for each one frame (1/60 seconds). For example, it is now assumed that the position of the object for (k-1) frame is PMk-1, the velocity is VMk-1, the acceleration is Amk-1, time for one frame is Δt. Thus, the position PMk and velocity VMk of the object for k frame can be determined by the following formulas (1) and (2):

$$PMk = PMk\text{-}1 + VMk\text{-}1 \times \Delta t \qquad (1)$$

$$VMk = VMk\text{-}1 + Amk\text{-}1 \times \Delta t \qquad (2)$$

The adjustment information setting section 114 is to set

(or prepare) an adjustment data used for adjusting the display property of the display section (or monitor) 190 such as brightness, color density, color tone, sharpness or the like, based on the operational data inputted by the player through the operating section (or game controller) 160.

The saving section 116 is to save the adjustment data set by the adjustment information setting section 114 (or data used for adjusting the brightness, color density, color tone, sharpness or the like) in the saved information storage device 194.

In this embodiment, the transformation relative to the image information of the original image is performed based on the adjustment data which is obtained by adjusting the display property or loaded from the saved information storage device 194. In this case, such a transformation will be realized by the function of an index-number setting section 134 or drawing section 140 (or texture mapping section 142), all of which will be described later.

The image generation section 130 comprises a geometry-processing section 132, an index-number setting section 134 and a drawing section 140.

The geometry-processing section 132 performs geometry-processing (or three-dimensional computation) such as coordinates transformation, clipping, perspective transformation, light-source calculation and so on. After subjected to the geometry-processing (or perspective transformation), the object data (such as shape data including

the vertex coordinates and others in the object, or vertex texture coordinates, brightness data and the like) is saved in a main storage region (or main memory) 172 in the storage section 170.

The index-number setting section 134 is to set the image information of the original image (e.g., perspective-transformed image information) as an index number in an LUT (lookup table) section 178 for an index color texture-mapping. The image information of the original image may take any of various information forms such as color information (RGB, YUV or the like), an alpha(α) value (or any information stored in association with each of the pixels, other than the color information), a depth value (Z value) and so on.

In this embodiment, an alpha value for each pixel in the original image is set at a value corresponding to the depth value for that pixel by performing the index color texture-mapping relative to the virtual object using the lookup table in which the depth value for each pixel in the original image has been set as an index number. Thus, a so-called depth of field can be represented according to this embodiment.

Relationship between a depth value (index number) and an alpha value for each pixel in the LUT is preferably set such that a pixel which is located farther from the focus of the virtual camera has a larger alpha value (or has a higher blending rate for a defocused image in a broad sense). In addition, the depth of field or defocusing effect may variably be controlled by changing the relationship between a depth value and an alpha

value for each pixel in the LUT.

The drawing section 140 is to draw the geometry-processed object (or model) in a drawing region 174 (which is a region in a frame buffer, an additional buffer or the like for storing the image information by a pixel unit). The drawing section 140 comprises a texture mapping section 142, an alpha(α)-blending section 144, a hidden surface removal section 146 and a masking section 148.

The texture mapping section 142 performs a process of mapping a texture stored in the texture storage section 176 onto an object (including a process of specifying a texture to be mapped on an object, a process of transferring a texture and other processes). In such a case, the texture mapping section 142 can perform the texture mapping using the index color texture-mapping LUT (lookup table) stored in the LUT storage section 178.

In this embodiment, the texture mapping section 142 performs the texture mapping relative to the virtual object (such as a polygon having a size equal to that of the display screen, a polygon having a size equal to that of a block of divided screen and the like) using the LUT in which image information of an original image is set as an index number. Thus, a process of transforming a depth value (Z value) into N bits, a process of transforming a depth value into an alpha value and other image transformation processing (e.g., video filtering such as gamma correction, negative/positive inversion, posterization, solarization, binarization, monotone filtering

and sepia filtering) can be realized with reduced processing load.

In addition, this embodiment implements the generation of a defocused image (or most-defocused image) blended with the original image through the alpha blending (such as narrow-sensed alpha blending, additive alpha blending, subtractive alpha blending, or translucency processing) by effectively utilizing texture mapping by texel interpolation method (bilinear or trilinear filtering).

More particularly, the texture mapping section 142 maps the original image set as a texture on the virtual object (which is an object having its shape equal to that of the defocused region) through the texel interpolation method while, for example, shifting the texture coordinates by a value smaller than one pixel (texel) (or shifting it from the texture coordinates obtained based on the drawn position of the original image). Thus, the defocused image to be blended with the original image can be generated through a simplified procedure by which the texture coordinates are only shifted.

The alpha-blending section 144 is to blend (combine) an original image with its defocused image based on an alpha value (A value) which has been set for each of the pixels in the drawing region 174 (a frame buffer, an additional buffer or the like). For example, when narrow-sensed alpha blending is performed, the original image is blended with the defocused image as represented by the following formulas:

$$R_0 = (1 - \alpha) \times R_1 + \alpha \times R_2 \quad (3)$$

$$G_0 = (1 - \alpha) \times G_1 + \alpha \times G_2 \quad (4)$$

$$B_0 = (1 - \alpha) \times B_1 + \alpha \times B_2 \quad (5)$$

where R_1, G_1 and B_1 are respectively color (or brightness) R, G and B components in the original image already drawn in the drawing region 174; R_2, G_2 and B_2 are respectively color R, G and B components in the defocused image; and R_0, G_0 and B_0 are respectively R, G and B components generated through the alpha blending.

The alpha-blending section 144 performs processing (such as additive alpha blending or alpha blending) for blending: transformed color information (R, G and B) obtained by setting the K-th color component (e.g., R component) in the original image as an index number in LUT; transformed color information obtained by setting the L-th color component (e.g., G component) as an index number in LUT; and transformed color information obtained by setting the M-th color component (e.g., B component) as an index number in LUT.

The hidden surface removal section 146 performs the hidden-surface removal according to Z-buffer algorithm, using a Z buffer (e.g., depth buffer or Z plane) 179 in which the Z value (depth value) is stored. In this embodiment, the Z value written in this Z buffer 179 is transformed into an alpha value on which the original image is blended with the defocused image.

The masking section 148 performs the masking such that when the color information is to be transformed by setting the

color component (e.g., R component) in the original image as an index number in the LUT, the other color components (e.g., G and B components) in the transformed color information will not be drawn in the drawing region (a frame buffer or an additional buffer). When the color information is to be transformed by setting the G component as an index number, the masking will be carried out relative to the R and B components. When the color information is to be transformed by setting the B component as an index number, the masking will be carried out relative to the R and G components.

The game system of this embodiment may be dedicated for a single-player mode in which only a single player can play the game or may have a multi-player mode in which a plurality of players can play the game.

If a plurality of players play the game, only a single terminal may be used to generate game images and sounds to be provided to all the players. Alternatively, a plurality of terminals interconnected through a network (transmission lien or communication line) may be used in the present invention.

2. Features of this embodiment

2.1 Use of index color texture-mapping

As described, the first technique shown in Fig. 1A cannot essentially realize the gamma correction (or video filtering) since it requires an excessive processing load on CPU. Furthermore, the second technique shown in Fig. 1B cannot realize the gamma correction in the domestic game system, since

it does not have such a gamma-correction circuit which is of dedicated hardware.

The inventor aimed at the presence of the lookup table, LUT, which was used in the index color texture-mapping.

More particularly, in the index color texture-mapping process, the index number rather than the actual color information (RGB) is stored in association with each texel in the texture as shown at A1 in Fig. 3, for saving the capacity of the texture storage section. The color information specified by the index number is also stored in the index color texture-mapping LUT (or color pallet), as shown at A2 in Fig. 3. When it is wanted to perform the texture mapping relative to an object, the LUT is referred to based on the index number for each texel in the texture to read the corresponding color information out of the LUT and to draw the read color information in the frame buffer.

The texture mapping in such an index color mode reduces the number of usable colors (e.g., 256 colors), compared with the texture mapping in the conventional modes using no LUT. However, the index color texture-mapping does not require the storage of the actual color information (e.g., 16-bit color information) in the texture storage section. This enables the capacity of the texture storage section to be greatly saved.

This embodiment is characterized by using such an index color texture-mapping process in a form different from the conventional forms.

As shown at B1 in Fig. 4, the image information (e.g.,

color information) of an original image drawn in a frame buffer (which is, in a broad sense, a drawing region) is first set as an index number in a gamma correction lookup table (LUT). In other words, this image information is considered as an index number. As shown at B2, the LUT in which the image information of the original image is set as an index number is then used to perform the index color texture mapping relative to a virtual object (e.g., a polygon having a size equal to that of the display screen) for transforming the image information of the original image. As shown at B3, the transformed image information is then drawn back to the frame buffer (or drawing region).

In such a manner, this embodiment successfully obtains such a gamma corrected image as shown in Fig. 5B from such an original image as shown in Fig. 5A. In other words, the image of Fig. 5B has a stronger contrast than the image of Fig. 5A.

For example, the first technique of Fig. 1A cannot increase its processing speed and can also increase its processing load on CPU since it must execute all the processes of reading the image information of the original image, referring to the gamma correction LUT and writing the color information back to the frame buffer in the software running on the CPU.

On the contrary, this embodiment can realize the gamma correction by effectively using the index color texture mapping which is executed by the dedicated hardware drawing processor (or drawing section) with increased speed. According to this embodiment, therefore, the gamma correction can be executed at

a speed higher than that of the first technique shown in Fig. 1A. It becomes easier that the gamma correction for the entire display screen is completed within one frame (e.g., 1/60 seconds or 1/30 seconds).

Since the index color texture mapping can be executed by the drawing processor operable independently of the main processor (CPU), the increase in the processing load on the main processor (CPU) can be minimized. As a result, the gamma correction will not adversely affect any other processing.

The conventional game systems could not very increase the capacity of the drawing processor. It was thus difficult that the drawing of the original image into the frame buffer as well as the drawing of the polygon having a size equal to that of the display screen are completed within one frame.

However, the game systems have been capable of using a drawing processor having its very high fill rate (or the number of texels renderable for one second) since the capacity of the drawing processor was highly improved in comparison with the capacities of the other circuit blocks. Therefore, the drawing of the original image into the frame buffer as well as the drawing of the polygon having a size equal to that of the display screen can easily be completed within one frame. Thus, the gamma correction can freely be realized by effectively using the index color texture-mapping.

In addition, the second technique of Fig. 1B leads to increase of the game system manufacturing cost since it separately requires a gamma-correction circuit which is of a

dedicated hardware. The domestic game systems not originally having such a gamma-correction circuit could not realize the second technique shown in Fig. 1B and but take the technique of Fig. 1A.

On the contrary, this embodiment realizes the gamma correction by effectively using the index color texture-mapping which is executed by the hardware originally possessed by the drawing processor. According to this embodiment, therefore, it is not required that such a gamma-correction circuit as shown in Fig. 1B is newly provided with the cost of the game system being minimized. Even in the domestic game system not originally having any gamma-correction circuit, the gamma correction can be realized through hardware with increased speed.

Although Fig. 4 has been described to realize the gamma correction (or video filtering) by performing the texture mapping on a polygon having a size equal to that of a display screen, the texture mapping may be executed on a polygon having a size equal to a size of a block obtained by dividing the display screen into blocks.

In other words, as shown at C1 in Fig. 6, the original image (or display screen) on the frame buffer is divided into blocks, each of which has an image to be texture mapped onto a polygon having a size equal to the size thereof using the LUT, as shown at C2. The resulting image having a size equal to that of the corresponding block is then drawn back to the frame buffer (or drawing region).

Alternatively, there may be generated such a polygon

that includes all or part of an object image perspective-transformed (or transformed into the screen coordinate system) and has its magnitude variable depending on that of the perspective-transformed object. The texture mapping will then be executed relative to such a polygon.

In such a manner, if a texture-mapped polygon is temporarily drawn in the additional buffer, for example, the area of VRAM occupied by the additional buffer can be reduced.

When the texture mapping is performed on a polygon having a size equal to that of the display screen as shown in Fig. 4, an additional buffer having a size equal to that of the display screen must be allocated on VRAM for temporarily drawing the polygon having a size equal to that of the display screen. This may adversely affect any other processing.

If the texture mapping is performed on a polygon having a size equal to a size of a block of divided screen as shown in Fig. 6, it is only required that an additional buffer having a size equal to a size of the block is provided on VRAM. Therefore, the area occupied by the additional buffer can be reduced. As a result, the limited hardware resource can effectively be utilized.

2.2 Various types of video filtering (LUT)

Fig. 7A shows the transformation property of the gamma correction.

In this figure, Bezier curve (which is, in a broad sense, free-surface curve) representing the transformation property

of the gamma correction is specified by four control points CP0, CP1, CP2 and CP3. In such a case, Y-coordinate of CP0 is set at Y0=0 while Y-coordinate of CP3 is set at Y3=255. The transformation property of the gamma correction can be adjusted by variably changing Y1 and Y2 which are Y-coordinates of CP1 and CP2, respectively.

The relational expression between input and output values X, Y in the gamma correction may be represented, for example, by the following formulas:

$$Y = Y20 + (X/255) \times (Y21 - Y20) \qquad (3)$$

where

$$Y20 = Y10 + (X/255) \times (Y11 - Y10)$$
$$Y21 = Y11 + (X/255) \times (Y12 - Y11)$$
$$Y10 = Y0 + (X/255) \times (Y1 - Y0)$$
$$Y11 = Y1 + (X/255) \times (Y2 - Y1)$$
$$Y12 = Y2 + (X/255) \times (Y3 - Y2)$$

When an index number is set at the input value X in the above formula (3) and when outputs ROUT, GOUT and BOUT for the respective color components are set at the output value Y, such a gamma correcting LUT as shown in Fig. 7B is provided. This LUT is then transferred to VRAM and used to perform the index color texture-mapping as described in connection with Fig. 4 or other figure. Thus, there can be obtained an image which is

formed by performing the video filtering of the gamma correction relative to the original image.

According to this embodiment, various types of video filtering other than the gamma correction may be realized relative to the original image on the frame buffer.

Fig. 8A shows the transformation property of a negative/positive inversion video filtering. The relational expression between input and output values X, Y in the negative/positive inversion may be represented by the following formula:

$$Y = 255 - X \qquad (4)$$

When an index number is set at the input value X in the above formula and when outputs ROUT, GOUT and BOUT for the respective color components are set at the output value Y, such a negative/positive inversion LUT as shown in Fig. 8B is provided. The resulting LUT is then used to perform the index color texture-mapping described in connection with Fig. 4 for obtaining the original image to which the negative/positive inversion video filtering is carried out.

Fig. 9A shows the transformation property of a posterization video filtering for representing a multi-gradation image while restricting some gradations. The relational expression between input and output values X, Y in the posterization may be represented by the following formula:

$$Y = \{INT\ (X/VAL)\} \times VAL \qquad (5)$$

where INT (R) is a function for rounding down the decimal point in R to provide an integer; and VAL is any value.

Fig. 9B shows the transformation property of a solarization video filtering for providing such an image effect that the inclination of a curve function between input and output values is inverted at a point. Moreover, Fig. 9C shows the transformation property of a binarizing video filtering for realizing the high-contrast effect in an image.

This embodiment may further realize monotone video filtering or sepia video filtering.

When color components prior to the monotone filtering are respectively RIN, GIN and BIN and when the color components after the monotone filtering are respectively ROUT, GOUT and BOUT, the transformation of the monotone filtering may be represented by the following formulas:

$$ROUT = 0.299 \times RIN + 0.587 \times GIN + 0.114 \times BIN \qquad (6)$$

$$GOUT = 0.299 \times RIN + 0.587 \times GIN + 0.114 \times BIN \qquad (7)$$

$$ROUT = 0.299 \times RIN + 0.587 \times GIN + 0.114 \times BIN \qquad (8)$$

The following formulas (9), (10) and (11) are defined herein as the relational expression between output values (ROUTR, GOUTR, BOUTR) relative to the input value RIN, the relational expression between output values (ROUTG, GOUTG, BOUTG) relative to the input value GIN and the relational

expression between output values (ROUTB, GOUTB, BOUTB) relative to the input value BIN, respectively.

$$(ROUTR, GOUTR, BOUTR) = (0.299 \times RIN, 0.299 \times RIN, 0.299 \times RIN) \quad (9)$$

$$(ROUTG, GOUTG, BOUTG) = (0.587 \times GIN, 0.587 \times GIN, 0.587 \times GIN) \quad (10)$$

$$(ROUTB, GOUTB, BOUTB) = (0.114 \times BIN, 0.114 \times BIN, 0.114 \times BIN) \quad (11)$$

Based on the above formulas (9), (10) and (11), such monotone filtering lookup tables LUTR, LUTG and LUTB as shown respectively in Figs. 10A, 10B and 11 are provided. These lookup tables LUTR, LUTG and LUTB are then used to perform the index color texture-mapping for obtaining the original image to which the monotone filtering is applied.

For the sepia filtering, the transformation thereof may be represented by the following formulas:

$$ROUT = 0.299 \times RIN + 0.587 \times GIN + 0.114 \times BIN + 6 \quad (12)$$

$$GOUT = 0.299 \times RIN + 0.587 \times GIN + 0.114 \times BIN - 3 \quad (13)$$

$$BOUT = 0.299 \times RIN + 0.587 \times GIN + 0.114 \times BIN - 3 \quad (14)$$

However, the clamping shall be carried out such that:

$$0 \leqq (ROUT, GOUT, BOUT) \leqq 255$$

With the sepia filtering, the following formulas may be defined.

$$(ROUTR, GOUTR, BOUTR) = (0.299 \times RIN+2,\ 0.299 \times RIN-1,\ 0.299 \times RIN-1) \quad (15)$$

$$(ROUTG, GOUTG, BOUTG) = (0.587 \times GIN+2,\ 0.587 \times GIN-1,\ 0.587 \times GIN-1) \quad (16)$$

$$(ROUTB, GOUTB, BOUTB) = (0.114 \times BIN+2,\ 0.114 \times BIN-1,\ 0.114 \times BIN-1) \quad (17)$$

Based on the above formulas (15), (16) and (17), sepia filtering lookup tables LUTR, LUTG and LUTB are provided. These lookup tables LUTR, LUTG and LUTB are then used to perform the index color texture-mapping for obtaining the original image to which the sepia filtering is applied.

2.3 Masking

The gamma correction requires such an LUT that outputs one value (ROUT, GOUT or BOUT) relating to one input value (RIN, GIN or BIN).

However, the index color texture mapping LUT as shown in Fig. 3 is not designed for gamma correction and will thus output a plurality of values (e.g., ROUT, GOUT and BOUT) relating to one input value (index number). There is thus a problem in that the mismatching for this LUT should be overcome.

If the image information of the original image (e.g., R, G, B, Z value or alpha value) is to be set at an index number

in the LUT, this embodiment performs a masking process in which only the necessary image information among the transformed image information is drawn in a drawing region (a frame buffer or an additional buffer) such that the other image information will not be drawn. If the image information of the original image is color information and when one color component in the original image is set at an index number in the LUT, the masking process is carried out such that the other color components after transformed will not be drawn in the drawing region.

More particularly, when an LUT in which the R plane value in the original image is set as an index number is used to perform the texture mapping as shown at D1 in Fig. 12, three plane values, R(ROUT), G(GOUT) and B(BOUT), will be outputted. In such a case, as shown at D2, only the outputted R plane values are drawn and the other G and B plane values will not be drawn in the drawing region through the masking process.

When the texture mapping was carried out after the G plane value in the original image had been set as an index number as shown at D3 in Fig. 12, only the outputted G plane value is drawn in the drawing region such that the other R and B plane values will not be drawn in the drawing region through the masking process, as shown at D4.

When the texture mapping was carried out after the B plane value in the original image had been set as an index number as shown at D5 in Fig. 12, only the outputted B plane value is drawn in the drawing region such that the other R and G plane values will not be drawn in the drawing region through the

masking process.

In such a manner, the index color texture-mapping LUT not originally designed for gamma correction can be used to execute the transformation of the original image, with reduced processing load.

2.4 Blending

The masking technique described in connection with Fig. 12 is also useful for any of various types of video filtering other than the gamma correction, such as negative/positive inversion, posterization, solarization, binarization and so on which have been described in connection with Figs. 8A to 9C.

On the contrary, it is desirable to take the following technique if it is wanted to realize the monotone or sepia video filtering. This technique is to blend the color information (R, G, B) obtained by setting the R component in the original image as an index number in the LUT, the color information (R, G, B) obtained by setting the G component as an index number in the LUT and the color information (R, G, B) obtained by setting the B component as an index number in the LUT.

More particularly, as shown at E1 in Fig. 13, three plane values, R(ROUTR), G(GOUTR) and B(BOUTR) as shown at E2 are obtained by setting the R (RIN) plane value in the original image as an index number and using such an LUT as shown in Fig. 10A for texture mapping. In this case, the relational expression between RIN and (ROUTR, GOUTR, BOUTR) may be represented by the above formula (9) or (15).

When the G (GIN) plane value in the original image is set at an index number and the LUTG of Fig. 10B is used to perform the texture mapping as shown at E3 in Fig. 13, three plane values, R(ROUTG), G(GOUTG) and B(BOUTG) as shown at E4 are obtained. In such a case, the relational expression between GIN and (ROUTG, GOUTG, BOUTG) may be represented by the above formula (10) or (16).

When the B (BIN) plane value in the original image is set at an index number and the LUTG of Fig. 11 is used to perform the texture mapping as shown at E5 in Fig. 13, three plane values, R(ROUTB), G(GOUTB) and B(BOUTB) as shown at E6 are obtained. In such a case, the relational expression between BIN and (ROUTB, GOUTB, BOUTB) may be represented by the above formula (11) or (17).

Furthermore, as shown at E7 in Fig. 13, the color information of R(ROUTR), G(GOUTR) and B(BOUTR) shown at E2, the color information of R(ROUTG), G(GOUTG) and B(BOUTG) shown at E4 and the color information of R(ROUTB), G(GOUTB) and B(BOUTB) shown at E6 are blended (or added) together.

In such a manner, the monotone or sepia video filtering can be realized as shown by the above transformation formulas (6), (7) and (8) or (12), (13) and (14).

2.5 Application for Z value and alpha value

Use of the color information R, G and B outputted based on the index color texture-mapping LUT has been described.

However, the alpha value (that is A-value information

other than the color information set in association with the pixels) outputted based on the index color texture-mapping LUT may be used.

For example, as shown in Fig. 14, the R (or G or B) plane value may be set as an index number in the LUT which is in turn used to perform the texture mapping for generating an alpha (or αOUT) plane. The resulting alpha plane may be used to perform the masking process or the like.

More particularly, an LUT is used in which an alpha value is set such that an alpha value (or an α OUT) becomes zero when an R value is 0 to 127 and that an alpha value becomes 255 when an R value is 128 to 255. The masking process will not be performed relative to a pixel having its alpha value smaller than 255 and will be carried out relative to a pixel having its alpha value equal to 255. Thus, the masking process will be performed only relative to pixel having their R values equal or larger than 128. As a result, the masking process may be carried out depending the magnitude of R value in each pixel.

The generated alpha plane value may be used as an alpha blending coefficient (transparency, translucency or opacity).

The image information set as an index number in the LUT is not limited to color information and may be one that is on the drawing region (VRAM) and can set as an index number in the LUT.

For example, as shown in Fig. 15, the Z value (or depth value) may be set as an index number in the LUT.

In such a case, the alpha plane value obtained by

performing the index color texture-mapping after the Z value has been set as an index number may be used as an alpha blending coefficient, for example. Thus, an alpha value can be set depending on the Z value so that the depth of field or the like can be represented by using a defocused image.

More particularly, such an LUT as shown in Fig. 15 is used to perform the texture mapping such that alpha values αA, αB, αC and αD for pixels A, B, C and D in an original image are set at values corresponding to Z values ZA, ZB, ZC and ZD for the respecting pixels A, B, C and D, as shown at F1 in Fig. 16. Thus, for example, such an alpha plane as shown at F2 in Fig. 16 may be generated. More particularly, an alpha value will be set to be increased as that a pixel corresponding to that an alpha value is farther from the focus of the virtual camera 10 (gazing point) (or a pixel has a larger difference between its Z value and the Z value of the focus). Thus, the rate of blending the original image with its defocused image increases as the pixel thereof is spaced farther apart from the focus of the virtual camera 10.

Next, as shown at F3 in Fig. 16, alpha blending of the original and its defocused images is carried out based on the generated alpha plane (an alpha value set for each pixel). Fig. 17A shows an original image while Fig. 17B shows its defocused image.

By performing alpha blending of the original image (Fig. 17A) and its defocused image (Fig. 17B) based on the alpha values set depending on the Z values (or depth values) in such a manner,

for example, an image can be generated in which the degree of defocusing therein will be increased as the pixels thereof are spaced farther apart from the focus of the virtual camera (that is, a focused point). This enables a so-called depth of field to be represented. Thus, unlike a conventional game image in which all objects in a screen are focused, this embodiment can generate a more realistic and natural image focused depending on the distance from a viewpoint like a real view. As a result, the player's feel for virtual reality can highly be improved.

Fig. 18 exemplifies the setting of an alpha value depending on a Z value. In this figure, the alpha value is normalized to have its magnitude equal to or less than 1.0.

In Fig. 18, the area is partitioned into regions AR0-AR4 and AR1' to AR4' depending on Z values Z1 to Z4 and Z1' to Z4' (threshold values). alpha values $\alpha 0$ to $\alpha 4$ and $\alpha 1'$ to $\alpha 4'$ are set relative to these regions AR0 to AR4 and AR1' to AR4'.

For example, a pixel located in the region AR1 between Z1 and Z2 may be set at $\alpha 1$; a pixel located in the region AR2 between Z2 and Z3 may be set at $\alpha 2$; a pixel located in the region AR1' between Z1' and Z2' may be set at $\alpha 1'$; and a pixel located in the region AR2' between Z2' and Z3' may be set at $\alpha 2'$.

The alpha values for the respective regions may be represented by the following relational expressions:

$$\alpha 0 < \alpha 1 < \alpha 2 < \alpha 3 < \alpha 4 \quad (18)$$

$$\alpha 0 < \alpha 1' < \alpha 2' < \alpha 3' < \alpha 4' \quad (19)$$

As will be apparent from these formulas, the alpha value is increased as the pixel is located farther from the focus of the virtual camera 10 (or gazing point). In other words, the alpha value is so set that the rate of blending between the original image and its defocused image is increased as the pixel has a larger difference between its Z value and the Z value of the focus of the virtual camera 10.

By so setting the alpha value, a more defocused image can be generated as the pixel is located farther apart from the focus of the virtual camera. This enables a so-called depth of field to be represented.

And yet, this embodiment is advantageous in that the processing load thereof is highly reduced since the Z value for each pixel can be transformed into the alpha value through only a single texture mapping using the LUT.

One of the alpha value setting techniques not using the LUT may be considered to be such a technique as shown in Figs. 19A, 19B and 19C.

As shown in Fig. 19A, the alpha value of a deeper pixel from a virtual object OB1 (or polygon) having its Z value set as Z1 is updated by drawing it in a frame buffer. In other words, the alpha value of the deeper pixel from the object OB1 is updated by effectively using the hidden-surface removal based on the Z value.

Next, as shown in Fig. 19B, the alpha value for a deeper pixel from a virtual object OB2 having its Z value set at Z2 is updated by drawing it in the frame buffer. Similarly, as shown

in Fig. 19C, the alpha value for a deeper pixel from a virtual object OB3 having its Z value set at Z3 is updated by drawing it in the frame buffer.

In such a manner, the alpha value of a pixel in the area AR1 can be set as $\alpha 1$; the alpha value of a pixel in the area AR2 can be set as $\alpha 2$; and the alpha value of a pixel in the area AR3 can be set as $\alpha 3$. In other words, the alpha value for each pixel can be set at a value corresponding to the Z value thereof.

However, this technique requires that the virtual object drawing process is repeated times corresponding to the number of threshold steps for Z value. For example, in Fig. 18, the drawing process should be repeated eight times. This is disadvantageous in that the drawing load is increased. On the contrary, if the number of threshold steps is reduced to relieve the drawing load, the boundary between the threshold Z values will be viewed as a stripe on the display screen, leading to reduction of the image quality.

According to the technique of this embodiment in which the Z value is transformed into the alpha value using the LUT, the Z values for all the pixels can be simultaneously be transformed into alpha values at a time. If the index number (entry) in the LUT is of 8 bits, the alpha values partitioned by 256 threshold Z value steps can be obtained. It can be prevented that the boundary between the threshold Z values will be viewed as a stripe on the display screen. Therefore, a high-quality image can be generated with reduced processing

load.

2.6 Formation of 8-bit Z value

To improve the accuracy of the hidden-surface removal in the game system, the number of bits in the Z value is very large, such as 24 bits or 32 bits.

On the other hand, the number of bits in the index number (entry) of the index color texture-mapping LUT is smaller than that of the Z value, such as 8 bits.

Where the Z value is to be transformed into the alpha value using such an index color texture-mapping LUT as shown in Fig. 15, therefore, a pre-process in which the Z value is transformed into a Z value having the same number of bits as that of the LUT index number is required. If the number of bits in the LUT index number is 8 bits, the Z value must be transformed into Z2 value (or second depth value) of 8 bits.

In this case, to obtain a consistent Z2 value, upper eight bits including the most-significant bit in the Z value must be used to form the Z2 value. In Fig. 20, eight bits ranging between bit 23 (or most-significant bit) and bit 16 is selected and used to form the Z2 value.

It has been found, however, that if the upper bits ranging between bit 23 and bit 16 in the Z value are used to form the Z2 value which is in turn transformed into the alpha value using the LUT, the number of partitions for alpha values will be reduced.

As shown in Fig. 21, for example, it is assumed that the

Z value is of 4 bits and that the upper two bits of the Z value are selected and used to form the Z2 value. In this case, if the Z2 values are transformed into alpha values, the alpha values will be partitioned into four steps by threshold values within all the range of Z = 0 to 15.

However, an object in front of a screen SC (perspective-transformed screen) as shown at OB1 in Fig. 21 will usually be near-clipped. If an object located in a range of Z = 10 to 15 in Fig. 21 is near-clipped, for example, it becomes rare that the most-significant bit in the Z value becomes one (1). In any event, the object as OB2 located near the screen SC is not required that its degree of defocusing is accurately controlled since it will necessarily be formed to be the most defocused image. Therefore, two steps of partitioning the portion as shown by G1 in Fig. 21 are unnecessary.

To avoid such a situation, this embodiment transforms the Z value (depth value) for each pixel in an original image into a Z2 value (second depth value) formed by lower bits I to J (e.g., bits 19 to 12) than the most-significant bit of the Z value. This Z2 value is set as an index number in the index color texture-mapping LUT and then performs the texture mapping for determining the alpha value for each pixel. Based on the determined alpha value for each pixel, alpha blending is performed on the original image and its defocused image.

In such a manner, as shown in Fig. 22, the degree of defocusing can accurately be controlled by using the alpha values which are partitioned with the multi-step thresholds

(four steps in Fig. 22) only relating to some objects (e.g., OB3 and OB4) that are located near the focus of the virtual camera 10 (or gazing point). Thus, the quality of the generated image can be improved.

In this embodiment, the Z2 value may be clamped at a given value depending on the bit values other than the Z value bits I to J. More particularly, as shown in Fig. 22, the Z2 value may be clamped at the maximum value (which is, in a broad sense, a given value) when the upper bit in the Z value becomes one (1), for example. Thus, the Z2 value will be set maximum for any object which is not required to control its degree of defocusing accurately, such as OB1 or OB2. A consistent image can be generated even though the Z value has been transferred into the Z2 value formed by its bits I to J.

2.7 Transformation of Z value using LUT

This embodiment realizes the process of transforming the Z value described in connection with Fig. 20 into the Z2 value (transformation into 8-bit form) by using the index color texture-mapping LUT for the texture mapping. In other words, the Z value is transformed into the Z2 value by setting the Z value as an index number in an LUT and using that LUT to perform the index color texture-mapping relative to a virtual object.

It is now assumed that a 24-bit Z value is transformed into a Z2 value using an LUT as shown in Fig. 23. In this case, as shown at H1 in Fig. 23, bits 15 to 8 (or M to N) in the Z value are set as index numbers in a first lookup table LUT1 which

is in turn used to perform the index color texture-mapping for transforming the Z value into Z3 value (or third depth value).

Next, as shown at H2 in Fig. 23, bits 23 to 16 (or K to L) in the Z value are set as index numbers in a second lookup table LUT2 which is in turn used to perform the index color texture-mapping for transforming the Z value into Z4 value (or fourth depth value).

Finally, as shown at H4 in Fig. 23, these Z3 and Z4 values are used to determine Z2 value which is in turn transformed into alpha value by using a third lookup table LUT3 as shown at H5.

More particularly, the Z3 value obtained by the transformation of LUT1 is drawn in a drawing region (a frame buffer or an additional buffer). Thereafter, the Z4 value obtained by the transformation of LUT2 is drawn in the drawing region. At this time, the Z2 value is determined while the lower four bits (or effective bits) in the Z3 value are masked so that these bits will not be overwritten by the Z4 value.

The technique of Fig. 23 enables any 8 bits (which are, in a broad sense, I to J) in the Z value to be fetched.

If it is wanted to fetch 8 bits in the Z value for setting them as index numbers in the LUT, only 8 bits in the Z value within a predetermined range, such as bits 23 to 16, 15 to 8 or 7 to 0, may be fetched.

On the other hand, the type of 8 bits to be fetched in the Z value is determined depending on the range of near-clipping or the focus position of the virtual camera (or gazing point), as described in connection with Figs. 20 and 21.

If only 8 bits of the Z value within a predetermined range, such as bits 23 to 16, 15 to 8 or 7 to 0, can be fetched, an appropriate alpha value for most accurately controlling the degree of defocusing near the focus of the virtual camera cannot be provided.

For example, it is now assumed that if bits 19 to 12 in the Z value are fetched as Z2 value, the effective number of partitions of the alpha value (or the number of threshold values of the Z value) can be stepped into 256. In such a case, if only bits 23 to 16 or 15 to 8 in the Z value can be fetched, the effective number of partitions in the alpha value will be stepped into only 16, leading to reduction of the image quality.

On the contrary, if the technique of Fig. 23 is used, any bits I to J in the Z value can be fetched as Z2 value even though only 8 bits in the Z value within a predetermined range can be fetched as described above. Thus, the threshold values of the Z value for partitioning the alpha value can optimally be set depending on the range of near-clipping or the focus position of the virtual camera and an image with its improved quality can be generated.

Figs. 24, 25 and 26A shows examples of LUT1, LUT2 and LUT3. Fig. 26B shows a curve of transformation property in the LUT3 for transforming Z2 value into alpha value.

As shown in Fig. 24, the LUT1 is used to shift bits 15 to 8 (or M to N) of the Z value inputted as index numbers rightward by four bits. For example, 0x10 and 0x20 (hexadecimal notation) may be transformed into 0x01 and 0x02, respectively.

As shown in Fig. 25, the LUT2 is used to shift bits 23 to 16 (or K to L) of the Z value inputted as index numbers leftward by four bits. For example, 0x01 and 0x02 may be transformed into 0x10 and 0x20, respectively.

If the inputted Z value is larger than 0x0F, the output of the LUT2 is clamped at 0xF0, as shown at Q1 in Fig. 25.

An example in which the Z value is clamped is shown in Fig. 27. Referring to Fig. 27, when bit 20 (or bit other than bits I to J) becomes one (1), the output of the LUT2 is clamped at 0xF0. Thus, Z2 value becomes 0xF1.

For example, if bits 19 to 12 in the Z value are directly fetched without clamping the output of the LUT2, the Z2 value will be 0x11, notwithstanding the bit 20 is one (1). This raises a problem in that the depth of field will wrongly be set.

The clamping of the LUT2 output can prevent such a problem and properly set the depth of field.

Moreover, the clamping of the LUT2 output will not provide any unnatural image since the degree of defocusing for the near-clipped object OB1 or the object OB2 located near the screen SC is only set maximum, as will be apparent from Fig. 22.

2.8 Generation of defocused image

This embodiment effectively uses the bilinear filtering type (or texel interpolation type) texture mapping to generate the defocused image (Fig. 17B) to be blended with the original image (Fig. 17A).

There may be produced a positional deviation between a pixel and a texel in the texture mapping.

In such a case, the point sampling type texture mapping renders the color CP (which is, in a broad sense, image information) at a pixel P (or sampling point) the color CA at a texel TA nearest the point P, as shown in Fig. 28.

On the other hand, the bilinear filtering type texture mapping provides the color CP of the point P which are interpolated by the colors CA, CB, CC and CD of texels TA, TB, TC and TD surrounding the point P.

More particularly, coordinate ratio in X-axis direction $\beta : 1-\beta$ $(0 \leqq \beta \leqq 1)$, and coordinate ratio in Y-axis direction $\gamma : 1-\gamma$ $(0 \leqq \gamma \leqq 1)$ are determined based on the coordinates of TA to TD and P.

In this case, the color CP of the point P (or the output color in the bilinear filtering type texture mapping) may be represented by the following formula:

$$CP = (1-\beta) \times (1-\gamma) \times CA + \beta \times (1-\gamma) \times CB + (1-\beta) \times \gamma \times CC + \beta \times \gamma \times CD \quad (20)$$

This embodiment aims at the fact that the bilinear filtering type texture mapping automatically interpolates the colors, thereby generating a defocused image.

More particularly, as shown at R1 in Fig. 29, an original image drawn in a frame buffer may be set as a texture. When this texture (or original image) is to be mapped on a virtual object

through the bilinear filtering type texture mapping process, texture coordinates given to the vertexes of the virtual object are shifted (dislocated or moved) rightward and downward, for example, by (0.5, 0.5). In such a manner, a defocused image in which the colors of the pixels in the original image spread into the surrounding pixels can automatically be generated through the bilinear filtering type interpolation.

If it is wanted to defocus the entire image, the shape of a virtual object onto which the texture (or original image) is mapped is set to be equal to the shape of a screen (or defocused region). For example, if the vertex coordinates of the screen are (X, Y) = (0, 0), (640, 0), (640, 480), (0, 480), the vertex coordinates of the virtual object are also (X, Y) = (0, 0), (640, 0), (640, 480), (0, 480).

In this case, if texture coordinates (U, V) given to the vertexes VX1, VX2, VX3 and VX4 of the virtual object are respectively set at (0, 0), (640, 0), (640, 480), (0, 480), the positions of the pixels in the screen will be identical with the positions of the texels in the texture. Therefore, the image will not be defocused.

On the contrary, if texture coordinates (U, V) given to the vertexes VX1, VX2, VX3 and VX4 of the virtual object are respectively set at (0.5, 0.5), (640.5, 0.5), (640.5, 480.5), (0.5, 480.5), the positions of the pixels in the screen will not be identical with the positions of the texels in the texture. Thus, the color interpolation will be executed to provide a defocused image through the bilinear filtering type

interpolation.

If it is wanted to defocus part of the screen, the shape of the virtual object may become equal to that of its defocused region.

As shown at R3 in Fig. 30, this embodiment a first defocused image by setting the original image as a texture and shifting it in the rightward and downward direction (first shift direction) by 0.5 for performing the bilinear filtering type texture mapping. Next, as shown at R4 in Fig. 30, the first defocused image is then set as another texture and shifted by 0.5 in the leftward and upward direction (second shift direction). The bilinear filtering type texture mapping is then executed to generate a second defocused image. Alternatively, the aforementioned procedure (or shifting in the rightward and downward direction and in the leftward and upward direction) may be repeated several times. Thus, a more natural and more defocused image can be generated.

The principle of generating the defocused image through the bilinear filtering type interpolation will be described below.

For example, it is now assumed as shown in Fig. 31A that the bilinear filtering type texture mapping is carried out after the texture coordinates are shifted by 0.5 texels in the rightward and downward direction. In this case, $\beta = \gamma = 1/2$ in the above formula (20). Therefore, if the colors in texels T44, T45, T54 and T55 are respectively made C44, C45, C54 and C55, the color CP44 of a pixel P44 may be represented by the

following formula:

$$CP44 = (C44 + C45 + C54 + C55)/4 \qquad (21)$$

As will be apparent from the foregoing, the color C44 of the texel T44 (which corresponds to the original color of the pixel P44 in the original image before transformation) will spread into the surrounding pixels P33, P34, P43 and P44 by each 1/4 through the transformation shown in Fig. 31A.

Thereafter, as shown in Fig. 31B, the image obtained in Fig. 31A is set as a texture, the coordinates of which are then shifted by 0.5 in the leftward and upward direction for performing the bilinear filtering type texture mapping. In this case, the pixels P33, P34, P43 and P44 in Fig. 31A will correspond to the texels T33, T34, T43 and T44 in Fig. 31B. The color C44 spread into the P33, P34, P43 and P44 (T33, T34, T43 and T44) by each 1/4 in Fig. 31A is then magnified further 1/4 and will spread into the four surrounding pixels. Eventually, the original color C44 of T44 will spread into the surrounding area by 1/16 = 1/4 × 1/4.

Thus, the color C44 (which corresponds to the original color of the pixel P44 in the original image drawn in the frame buffer) will spread into the pixels P33, P34 and P35 respectively by 1/16, 2/16 and 1/16 through the transformations of Figs. 31A and 31B. Furthermore, the color C44 will spread into the pixels P43, P44 and P45 respectively by 2/16, 4/16 and 2/16 and into the pixels P53, P54 and P55 respectively by 1/16,

2/16 and 1/16.

As a result, such a plane filter as shown in Fig. 32A will be applied to the original image through the transformations of Figs. 31A and 31B. Such a plane filter can uniformly spread the color of each pixel in the original image into the surrounding area. This can generate an ideal defocused image of the original image.

If the set of transformations in Figs. 31A and 31B are repeated two times, such a plane filter as shown in Fig. 32B will be applied to the original image. Such a plane filter can generate a more ideal defocused image than that of Fig. 32A.

2.9 Adjustment of monitor brightness

In a certain RPG or horror game in which a player controls a character on a screen to seek a dungeon, the brightness of the game image is usually set darkly for the purpose of causing the player to feel the dark atmosphere in the dungeon.

In such a case, if the brightness of the monitor on which the game image is displayed is set biased to the darkness, the contrast in the game image will become lower. This raises a problem in that the details of the shape or stripe in the dungeon will not clearly be looked or that a game image different from that intended by the game developer will be displayed.

To overcome such a problem, one technique may be considered in which a brightness adjustment button 22 on a monitor 20 is controlled directly by a player to adjust the brightness in the entire screen, as shown in Fig. 33.

However, such a technique is not convenient for the player since he or she must manually control the adjustment button 22 to adjust the brightness of the screen.

If the brightness of the monitor 20 has been adjusted for that game, the player must re-adjust the brightness for seeing any other image source (TV tuner, video or other game) after the previous game has been terminated.

Thus, this embodiment has such a purpose that a player can use a game controller 30 to set an adjustment data for adjusting the brightness of the monitor 20 (which is, in a broad sense, the displayed property), as shown in Fig. 34.

In Fig. 34, for example, the adjustment data may be set to increase the brightness on the entire screen when the player operates a cross key 32 on the game controller 30 to indicate the leftward direction and to decrease the brightness on the entire screen when the player operates a cross key 32 on the game controller 30 to indicate the rightward direction.

The set adjustment data is then saved in a saved information storage device 40 for storing the personal data (or saving data) of the player.

This embodiment performs the transformation relative to the image information of the original image based on the adjustment data which has been obtained by the adjustment of brightness (or displayed property) or loaded from the saved information storage device 40.

More particularly, the gamma-correction transformation property (Fig. 7A) is specified based on the adjustment data.

This specified transformation property is then used to prepare the gamma correction LUT (Fig. 7B) which is in turn used to perform the transformation relative to the original image through such a technique as described in connection with Fig. 4 or other.

In such a manner, the player can adjust the brightness of the monitor 20 using the game controller 30 without operation of the brightness adjustment button 22.

The adjustment data stored in the saved information storage device 40 is only effective for the game at which the player has carried out such an adjustment. If the player is to see any other image source after the previous game has terminated, therefore, the brightness will not be required to be restored. If the previous game is to be again played, the brightness of the monitor 20 will be adjusted based on the adjustment data loaded from the saved information storage device 40. Therefore, the re-adjustment of the brightness is not required. The convenience for the player can highly be improved.

Since the capacity of the saved information storage device 40 is limited, it is desirable that the adjustment data to be saved is as small as possible.

When the adjustment data for specifying, for example, the transformation property of gamma correction is to be saved, it is desirable that the control point data in a Bezier curve for representing the transformation property of gamma correction (which is, in a broad sense, a free-surface curve)

is saved in the saved information storage device 40. For example, Y coordinates for the control points CP0, CP1, CP2 and CP3 in Fig. 7A or only Y coordinates for the control points CP1 and CP2 may be saved. In such a manner, the storage capacity necessary to save the adjustment data can be saved with the remaining storage capacity being used for the other applications.

When the remaining storage capacity of the saved information storage device 40 is relatively large, all the contents of the gamma correction LUT shown in Fig. 7B may be saved in the saved information storage device 40.

3. Process of this embodiment

The details of the process of this embodiment will be described in connection with flowcharts shown in Figs. 35 to 39.

Fig. 35 is a flowchart illustrating a process of this embodiment in which the technique of Fig. 12 is taken.

First of all, such a transformation LUT as shown in Fig. 7B has been transferred to VRAM (step S1).

Next, the R plane value in the original image on the frame buffer is set as an index number in the LUT, as described at D1 in Fig. 12. This LUT is then used to apply the texture mapping to a polygon having a size equal to that of the display screen. This polygon is then drawn in an additional buffer (step S2). At this time, the other G and B values have been masked, as described at D2 in Fig. 12.

Next, as described at D3 in Fig. 12, the G plane value in the original image on the frame buffer is set as an index number in the LUT which is in turn used to apply the texture mapping to a polygon having a size equal to that of the display screen. This polygon is then drawn in an additional buffer (step S3). At this time, the other R and B values have been masked as described at D4 in Fig. 12.

Next, as described at D5 in Fig. 12, the B plane value in the original image on the frame buffer is set as an index number in the LUT which is in turn used to apply the texture mapping to a polygon having a size equal to that of the display screen. This polygon is then drawn in an additional buffer (step S4). At this time, the other R and G values have been masked as described at D6 in Fig. 12.

Finally, the images drawn in the additional buffers are drawn in the frame buffer through the texture mapping or the like (step S5).

Fig. 36 is a flowchart illustrating a process of this embodiment which takes the technique of Fig. 13.

First of all, such LUTR, LUTG and LUTB as shown in Figs. 10A, 10B and 11 are prepared and previously transferred to VRAM (step S10).

Next, as described at E1 in Fig. 13, the R plane value in the original image on the frame buffer is set as an index number in the LUTR. This LUTR is then used to apply the texture mapping to a polygon having a size equal to that of the display screen. This polygon is then drawn in a first additional buffer

(step S11).

Next, as described at E3 in Fig. 13, the G plane value in the original image on the frame buffer is set as an index number in the LUTG which is in turn used to apply the texture mapping to a polygon having a size equal to that of the display screen. This polygon is then drawn in a second additional buffer (step S12).

Next, as described at E5 in Fig. 13, the B plane value in the original image on the frame buffer is set as an index number in the LUTB which is in turn used to apply the texture mapping to a polygon having a size equal to that of the display screen. This polygon is then drawn in a third additional buffer (step S13).

Next, the image drawn in the first additional buffer is drawn in the frame buffer (step S14). The image drawn in the second additional buffer is then additionally drawn in the frame buffer (additive alpha blending) (step S15). Finally, the image drawn in the third additional buffer is additionally drawn in the frame buffer (step S16).

Figs. 37 and 38 show a flowchart for a process of transforming the Z value of the original image into alpha value which is in turn used to blend the original image with its defocused image (see Fig. 16).

First of all, the original image (or perspective-transformed image) is drawn in the frame buffer (step S21). At this time, Z value for each pixel will be written into the Z buffer.

Next, LUT1 (Fig. 24) for transforming bits 15 to 8 in the Z value of the Z buffer, LUT2 (Fig. 25) for transforming bits 23 to 16 and LUT3 (Fig. 26A) for transforming the Z value of 8-bit form into alpha value (A value) are transferred to VRAM (step S22).

Bits 15 to 8 in the Z value is set as an index number in the LUT1 which is in turn used to perform the texture mapping relative to a virtual polygon. This polygon is then drawn in an additional buffer (step S23).

Bits 23 to 16 in the Z value is set as an index number in the LUT2 which is in turn used to perform the texture mapping relative to a virtual polygon. This polygon is then drawn in an additional buffer (step S24). At this time, four lower bits (or data effective bits) in the Z value of 8-bit form have been masked for avoiding any overwriting.

Next, Z2 value of 8-bit form obtained at the step S24 is set as an index number in the LUT3 which is in turn used to perform the texture mapping relative to a virtual polygon. This polygon is drawn in the frame buffer (alpha plane) (step S25).

Next, the original image drawn in the work buffer at the step S21 is mapped on a virtual polygon through the bilinear filtering type interpolation while shifting the texture coordinates U and V by (0.5, 0.5). This virtual polygon is drawn in an additional buffer (step S26).

Next, the image drawn in the additional buffer at the step S26 is mapped on a virtual polygon through the bilinear filtering type interpolation while shifting the texture

coordinates U, V by (-0.5, -0.5). This polygon is drawn in the frame buffer (step S27). At this time, the alpha blending is carried out by using the alpha value drawn in the frame buffer at the step S25 to blend the original image with its defocused image.

In such a manner, a so-called depth of field can be represented.

Fig. 39 is a flowchart for illustrating a process of adjusting the brightness as described in connection with Fig. 34.

First of all, it is judged whether or not the brightness adjustment data exists in a memory card (or saved information storage device) and also whether or not the brightness adjustment data is to be loaded (step S30). If it is judged that the adjustment data should be loaded, it will be loaded from the memory card (step S31).

If it is judged that the adjustment data should not be loaded, it is judged that the player has selected an option screen for the brightness adjustment (display screen of Fig. 34) (step S32). If not so, the brightness adjustment data is set at a previously provided initial value (step S33). On the other hand, if the player has selected the option screen, the brightness adjustment data is set (prepared) based on the operational data from the player as described in Fig. 34 (step S34). The set brightness adjustment data is then saved in the memory card (step S35).

Next, the game image is dynamically transformed for each

frame, based on the resulting brightness adjustment data (initial adjustment data, set adjustment data or loaded adjustment data), through the technique as described in connection with Fig. 4 or other figure (step S36).

4. Hardware configuration

Hardware configuration for implementing this embodiment is shown in Fig. 40.

A main processor 900 operates to execute processing such as game processing, image processing, sound processing and other types of processing according to a program stored in a CD (information storage medium) 982, a program transferred through a communication interface 990 or a program stored in a ROM (information storage medium) 950.

A coprocessor 902 is to assist the processing of the main processor 900 and has a product-sum operator and analog divider which can perform high-speed parallel calculation to execute a matrix (or vector) calculation at high speed. If a physical simulation for causing an object to move or act (motion) requires the matrix calculation or the like, the program running on the main processor 900 instructs (or asks) that processing to the coprocessor 902.

A geometry processor 904 performs a geometry processing such as coordinate transformation, perspective transformation, light source calculation, curve formation or the like and has a product-sum operator and analog divider which can perform high-speed parallel calculation to execute a matrix (or vector)

calculation at high speed. For example, for the coordinate transformation, perspective transformation or light source calculation, the program running on the main processor 900 instructs that processing to the geometry processor 904.

A data expanding processor 906 performs a decoding process for expanding image and sound compressed data or a process for accelerating the decoding process in the main processor 900. In the opening screen, intermission screen, ending screen or other game screen, thus, an MPEG compressed animation may be displayed. The image and sound data to be decoded may be stored in the storage devices including ROM 950 and CD 982 or may externally be transferred through the communication interface 990.

A drawing processor 910 is to draw or render an object constructed by primitive surfaces such as polygons or curved faces at high speed. On drawing the object, the main processor 900 uses a DMA controller 970 to deliver the object data to the drawing processor 910 and also to transfer a texture to a texture storage section 924, if necessary. Thus, the drawing processor 910 draws the object in a frame buffer 922 at high speed while performing a hidden-surface removal by the use of a Z-buffer or the like, based on the object data and texture. The drawing processor 910 can also perform alpha blending (or translucency processing), depth cueing, mip-mapping, fogging, bilinear filtering, trilinear filtering, anti-aliasing, shading and so on. As the image for one frame is written into the frame buffer 922, that image is displayed on a display 912.

A sound processor 930 includes any multi-channel ADPCM sound source or the like to generate high-quality game sounds such as BGMs, sound effects and voices. The generated game sounds are outputted from a speaker 932.

The operational data from a game controller 942, saved data from a memory card 944 and personal data may externally be transferred through a serial interface 940.

ROM 950 has stored a system program and so on. For an arcade game system, the ROM 950 functions as an information storage medium in which various programs have been stored. The ROM 950 may be replaced by any suitable hard disk.

RAM 960 is used as a working area for various processors.

DMA controller 970 is to control the DMA transfer between processors and memories (RAM, VRAM, ROM and so on).

CD controller 980 drives a CD (information storage medium) 982 in which the programs, image data or sound data have been stored and enables these programs and data to be accessed.

The communication interface 990 performs data transfer between the image generating system and any external instrument through a network. In such a case, the network connectable with the communication interface 990 may take any of communication lines (analog phone line or ISDN) or high-speed serial bus. The use of the communication line enables the data transfer to be performed through the internet. If the high-speed serial interface bus is used, the data transfer may be carried out between the other game systems.

All the means of the present invention may be realized

(executed) only through hardware or only through a program which has been stored in an information storage medium or which is distributed through the communication interface. Alternatively, they may be executed both through the hardware and program.

If all the means of the present invention are executed both through the hardware and program, the information storage medium will have stored a program for realizing the respective means of the present invention through the hardware. More particularly, the aforementioned program instructs the respective processors 902, 904, 906, 910 and 930 which are hardware and also delivers the data to them, if necessary. Each of the processors 902, 904, 906, 910 and 930 will execute the corresponding one of the means of the present invention based on the instruction and delivered data.

Fig. 41A shows an arcade game system to which this embodiment is applied. Players enjoy a game by controlling levers 1102 and buttons 1104 while viewing a game images displayed on a display 1100. A system board (circuit board) 1106 included in the game system includes various processors and memories which are mounted thereon. An information (program or data) for realizing all the means of the present invention has been stored in a memory 1108 on the system board 1106, which is an information storage medium. Such information will be referred to "stored information" later.

Fig. 41B shows a domestic game system to which this embodiment is applied. A player enjoys a game by manipulating

game controllers 1202 and 1204 while viewing a game picture displayed on a display 1200. In such a case, the aforementioned stored information pieces have been stored in DVD 1206 and memory cards 1208, 1209 which are detachable information storage media in the game system body.

Fig. 41C shows an example wherein this embodiment is applied to a game system which includes a host machine 1300 and terminals 1304-1 to 1304-n (game devices or portable telephones) connected to the host machine 1300 through a network (which is a small-scale network such as LAN or a global network such as INTERNET) 1302. In such a case, the above stored information pieces have been stored in an information storage medium 1306 such as magnetic disk device, magnetic tape device, semiconductor memory or the like which can be controlled by the host machine 1300, for example. If each of the terminals 1304-1 to 1304-n are designed to generate game images and game sounds in a stand-alone manner, the host machine 1300 delivers the game program and other data for generating game images and game sounds to the terminals 1304-1 to 1304-n. On the other hand, if the game images and sounds cannot be generated by the terminals in the stand-alone manner, the host machine 1300 will generate the game images and sounds which are in turn transmitted to the terminals 1304-1 to 1304-n.

In the arrangement of Fig. 41C, the means of the present invention may be decentralized into the host machine (or server) and terminals. The above information pieces for realizing the respective means of the present invention may be distributed

and stored into the information storage media of the host machine (or server) and terminals.

Each of the terminals connected to the network may be either of home or arcade type. When the arcade game systems are connected to the network, it is desirable that each of the arcade game systems includes an saved information storage device (memory card or portable game machine) which can not only transmit the information between the arcade game systems but also transmit the information between the arcade game systems and the domestic game systems.

The present invention is not limited to the above-described embodiment, and various modifications can be made.

For example, part of the structural requirements in any of claims having dependent claims of the invention can be omitted. Primary part of an independent claim of the invention can be made dependent on any other independent claim.

The image information set as the index number in the index color texture-mapping lookup table is particularly desirable to be the information described in connection with this embodiment, but the present invention is not limited to such information.

The image transformation (or video filtering) realized by the present invention is not limited to those described in connection with Figs. 7A to 11.

Although it is particularly desirable that the technique of transforming the image information in such a form as is to save the adjustment information in the saved information

storage device is that described in connection with Fig. 4, the present invention is not limited to such a technique. The image information may be transformed through the techniques described in connection with Figs. 1A and 1B, for example.

Although it is particularly desirable that the transformation of the depth value into the second depth value is realized through the technique of Fig. 23 using the index color texture-mapping lookup table, such a transformation may be realized through any other technique.

The transformation property of the lookup table used to transform the depth value is not limited to any of such transformation properties as shown in Figs. 24, 25, 26A and 6B, but may be carried out in any of various other forms.

Although it is particularly desirable that the defocused image to be blended with the original image is generated through the technique described in connection with Figs. 29 and 30, the present invention is not limited to such a technique. For example, the defocused image may be generated by blending the original image with its shifting image or by blending an original image on the present frame with an original image on the previous frame.

The invention of generating the defocused image through the texel interpolation is not limited to the technique described in connection with Figs. 29 to 32B. For example, an area smaller than a screen may be set in which an original image will be defocused, rather than defocusing of the entire screen.

Although this embodiment has been described as to the

depth value increasing as the pixel thereof is spaced nearer away from the viewpoint, the depth value may be increased as the pixel thereof is spaced farther away from the viewpoint.

The present invention may similarly be applied to any of various other games such as fighting games, shooting games, robot combat games, sports games, competitive games, role-playing games, music playing games, dancing games and so on.

Furthermore, the present invention can be applied to various image generating systems such as arcade game systems, domestic game systems, large-scaled multi-player attraction systems, simulators, multimedia terminals, image generating systems, game image generating system boards and so on.